一种第三方模块适配接入视听媒体内容资源服务平台的技术研究

付 坤 邓成胜 倪业鹏
（中国传媒大学 数据科学与智能媒体学院，北京 100000）

摘 要：当前视听媒体领域普遍存在着技术架构不能适应发展需求的问题，一方面广播电视媒体不能满足互联网化业务的快速迭代与多样化服务发展需求，另一方面互联网视听媒体企业又在内容生产和播出环节中难以达到广电高质量内容生产与管理的要求。为此，国内外主流媒体及互联网视听媒体企业都在积极探索适应媒体转型需求的技术支撑架构体系。在此背景下，视听媒体行业从传统的单体架构逐渐向微服务架构转化。本研究通过分析当前视听媒体内容资源服务平台目前面临的扩展性差、原生平台开发功能不全、平台维护能力有限等问题，探索能够为视听媒体内容资源服务平台及该平台以外的第三方模块连接的新技术和新方案。

关键词：微服务；第三方模块；适配接入；容器；视听媒体 **中图分类号：**TP936 **文献标识码：**A
文章编号：1671-0134（2022）04-133-05 DOI：10.19483/j.cnki.11-4653/n.2022.04.040
本文著录格式：付坤，邓成胜，倪业鹏.一种第三方模块适配接入视听媒体内容资源服务平台的技术研究[J].中国传媒科技，2022（04）：133-137.

1. 研究背景

在“互联网+”的技术和业务创新大潮下，传统广电需要吸纳云计算和云原生技术，以及其底层所支撑的微服务技术。[1] 与传统技术架构相比，基于微服务技术架构的视听媒体内容资源服务平台具有明显优势，但也由于需要接入大量第三方微服务模块，平台存在着诸多问题，如：第三方微服务模块接口不规范、存在安全隐患、效率低下等问题；微服务架构系统和传统架构系统形成生态隔阂，不同架构的系统不能互用；微服务模块开发语言多样繁杂，导致系统的维护成本增加，不利于行业发展等问题。[2]

针对上述问题，本文提出了一种第三方模块适配接入视听媒体内容资源服务平台的技术方案，为视听媒体微服务生态建设发展提供了一条新的思路：将第三方模块（即除原生开发团队开发的功能模块以外，其他团队开发的微服务模块或非微服务模块）通过该技术进行适配接入之后，与原生平台相连，将第三方模块部署到原生平台上，以拓展平台的功能，进而形成视听媒体微服务生态系统。[3-4]

2. 整体解决方案

在计算网络技术的不断发展和普及下，软件体系架构在信息系统的整体设计中愈发重要，传统单体架构设计模式具有耦合度高、业务模块复用性差等问题，无法满足现代用户的使用需求。为了解决第三方模块适配接入尤其是单体架构应用适配接入的问题，本文提出第三方模块适配接入技术整体解决方案，如图1所示，该方案使用一种基于容器的微服务架构，通过使用容器技术保证第三方模块的可用性，提高适配接入效率。[5]

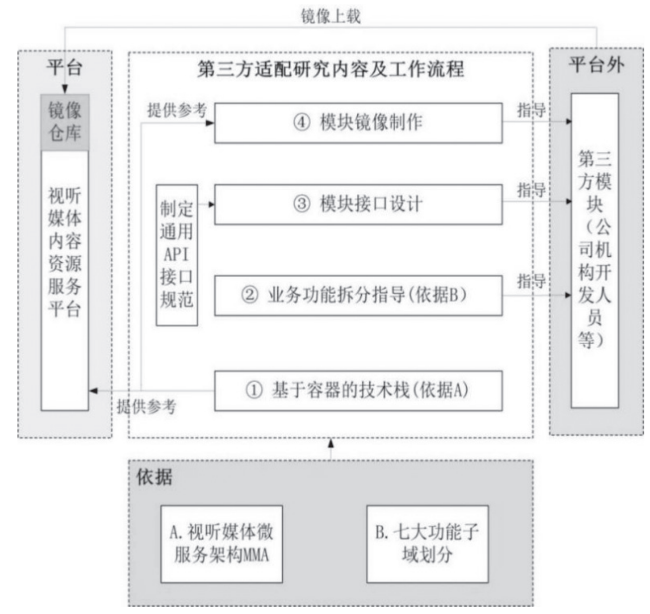


图1 第三方适配接入技术整体解决方案

通过对国内外相关技术进行调研，参考视听媒体微服务架构（MMA），提出基于容器的微服务技术栈，明确容器和镜像仓库是第三方模块适配接入的基础。依据

基金项目：国家重点研发计划“视听媒体微服务关键技术研究与应用”项目编号：2019YFB1405900

视听媒体七大功能子域划分，研究视听媒体业务微服务业务功能拆分方法和原则，为第三方模块开发团队提供指导，制定通用 API 接口规范。[6] 根据通用 API 接口规范，规范指导第三方模块开发团队进行微服务模块接口设计，为平台和复杂微服务接口提供统一交互界面。研究微服务模块镜像制作技术，指导第三方模块开发团队打包微服务模块代码和运行环境并上载至平台镜像仓库，使微服务可以跨平台部署上线运行。[7]

视听媒体微服务第三方模块适配接入技术整体解决方案，可以分为如下步骤。

2.1 业务的微服务化

通过对视听媒体业务和数据流，代码逻辑的提取，结合 7 大域 340 多个微服务划分，将原有的业务按需拆分成独立的、适合系统的微服务架构。[8-9]

2.2 微服务接口设计

模块拆分完成后，在业务系统背景下，根据微服务模块的功能和特性进行接口设计。指导接入方修改原有的单体应用结构或者不符合规则的接口，按照商议好的原则进行新的接口代码实现。[10]

2.3 微服务镜像打包封装

第三方开发团队本地自测完新的代码后，将第三方模块的代码和环境通过本方案提供的方法，创建对应的镜像打包封装，并在通过鉴权等一系列认证后，准入到镜像仓库中。

根据以上解决方案，本技术研究内容主要分为三个部分。

首先容器是微服务的载体 [11]，通过镜像封装技术可以把微服务架构的第三方模块打包后上载到镜像仓库，方便部署，因此第一部分研究内容为基于容器的微服务平台，主要研究适合第三方模块适配接入的技术栈。

其次对于单体架构的第三方模块，需要进行微服务化，微服务化主要包括服务拆分和接口设计，因此第二部分研究内容为单体架构第三方模块的微服务化。

对于微服务化后的第三方模块，需要以容器的方式，进行镜像封装，封装后的镜像可以屏蔽底层技术差异，在上载到镜像仓库后，模块可以跨平台运行，因此第三部分研究内容为第三方微服务模块镜像封装技术。[12]

以上三部分内容即完成了视听媒体微服务第三方模块适配接入技术研究，使单体架构或微服务化的第三方模块平滑接入平台。

3. 基于容器的微服务平台技术栈

根据对视听媒体微服务架构（MMA）和容器等技术的理解，本文提出了一种基于容器的微服务平台技术栈，为视听媒体内容资源服务平台建设提供参考。根据研究，目前只有基于容器的微服务技术栈可以保证第三方模块的适配接入，技术栈的具体方案如图 2 所示。

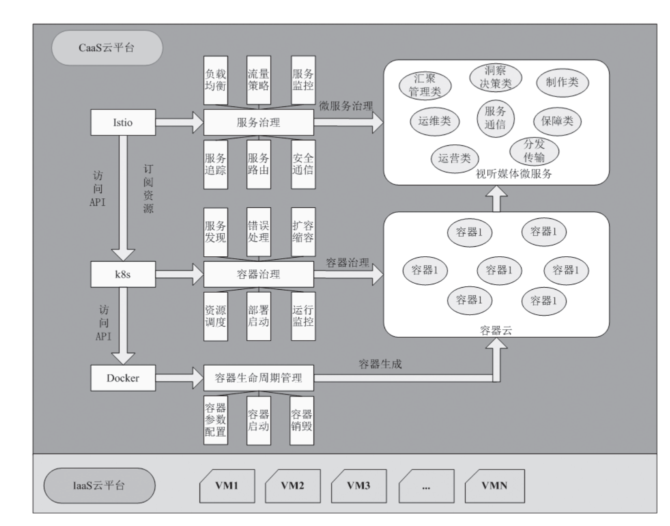


图 2　基于容器的微服务技术栈

整个平台分为两层，底层同样为 IaaS 层，可以是虚拟机或物理机，容器无须关心底层资源的实现形式。中间为 CaaS 层，主要提供容器级别的服务。CaaS 中包含了 Docker、k8s 和 Istio，分别对应容器生命周期管理，容器治理、微服务治理。该技术栈符合视听媒体微服务架构的特性需求，同时提供容器相关的服务，为第三方模块适配接入奠定基础。

4. 单体架构第三方模块微服务化

4.1 数据拆分方法

本文主要采取的数据拆分方法为优化的数据流驱动的拆分方法。具体拆分方法如图 3 所示：

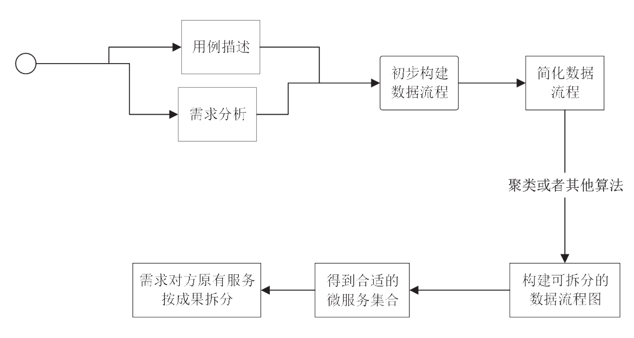


图 3　第三方模块拆分流程

①需求分析。由需求分析人员对待拆分的软件系统需求进行分析。根据系统用例或自然语言描述的业务逻辑，识别领域上下文和实体，为数据流图的构建做准备。

②精简数据流图构建。基于第①步中的用例和业务逻辑分析，由数据分析人员手动构建详细版本的分层数据流图和精简的数据流图。详细版本的数据流图构建需要遵循方法预设的分解规则，精简数据流图所构造的仅关注操作与相关数据存储之间的关系，排除了诸如外部实体以及数据流传输的具体数据之类的无关信息。[13]

③可拆分数据流图构建。从精简化数据流图中提取操作和数据存储之间数据交互信息，将精简数据流图通过算法自动转化可拆分的数据流图。之所以关注操作和数据存储之间的数据交互，目的是避免数据存储在拆分过程中被分到不同的微服务中，这样可以保证在减小微服务粒度的同时，尽可能减少不必要的数据一致性的问题。[14-15]

④候选微服务识别。对第③步导出的可拆分的数据流图，通过设计的算法进行拆分，具体是对同一数据存储的相关操作进行聚合，然后对出现重复操作的聚合结果进行合并，合并后的结果作为候选微服务。[16]

4.2 微服务划分算法优化

4.2.1 基于最小生成树的拆分 [17]

本文实现了基于最小生成树的算法来完成业务操作的聚类拆分，该算法依赖的参数是根据收集的操作间的数据流信息生成的带权图 G、预期的微服务个数 m 和每个微服务的类个数阈值 n。其中，图 G 的结点表示的是类级别的业务操作；边权重表示的是操作结点之间的交互频率，边权重值越大，表示操作之间的交互越频繁，耦合程度越高，即越倾向于被分到同一个微服务中。该阶段算法的基本原理是：首先，通过 Kruskal 算法生成图 G 的最小生成树 MST；然后对最小生成树的边权重进行分析和删除，以实现图的拆分。由于 G1 的边权重越大表示结点间的距离越近，因此在计算最小生成树之前需要对边权重值取倒数；所生成的 MST 中包含了关系最为紧密的业务操作结点和边，对 MST 涉及的边权重值进行排序和反转操作，依次删除权重值越高的边来实现耦合度较低的业务操作的拆分，并通过深度优先搜索算法（Depth-First-Search，简称 DFS）遍历剩余边组成的子图。本文通过预期的微服务个数和每个微服务类个数阈值这两个参数作为算法的终止条件，不同的参数设置可以产生不同的微服务化拆分结果，用户可以根据自己的需求进行设置，一定程度上保证了方法在使用过程中的灵活性。

4.2.2 基于 K-means 的数据库表拆分 [18]

Database per Service 原则倡导每个服务维护自己的数据库，其他的微服务通过向外暴露的接口来访问服务私有的数据，因此在向微服务架构迁移时，需要对数据库表进行拆分。现有的方法中涉及了数据库表的拆分，但与业务操作拆分类似，其仅考虑了操作与数据存储之间有无数据交互关系，使用预先定义的简单规则来实现拆分；其他工作集中在通过设计阶段数据库实体的结构分析以及其被系统访问的特征来进行拆分。然而一般情况下，单体系统，尤其是复杂单体系统中使用的集中式数据库中，数据库表之间存在耦合和粘连性，设计阶段单一特征分析和简单的规则无法全面解决该问题，有可能会造成拆分的粒度过细或过粗，从而导致后期微服务之间的频繁交互而影响性能。

本文在前一阶段业务拆分结果的基础上，依据收集的类级别的业务操作与数据存储之间的交互关系来分析其在运行时对数据的访问特征，通过 K-means 算法来实现数据库表的有效拆分。K-means 算法依赖的参数是上一阶段聚类产生的业务操作集合为结点、集合中操作与数据库表交互关系为边所生成的带权图 G，以及根据数据库实体关系分析初步设定的数据库表中心点集合。具体实现过程中，由于操作对数据的访问频率越高同样代表其之间的关系越紧密，因此以图 G 边权重值倒数计算结点到中心点的 Dijkstra 最短路径，将结点放到与之路径最短的中心结点簇中；然后在每个中心点的簇中选出结点度最大的结点为该簇的中心点，直至中心点不再发生变化或迭代次数为 0；最后输出中心点与其簇中结点，其代表了包含数据库表的微服务候选集。

4.3 第三方模块通用接口设计规范

视听媒体业务被拆分成 7 大域，340 多个微服务，这些微服务功能各不相同，本文结合面向对象编程和操作系统函数的设计思想，提出视听媒体微服务标准化接口设计方案如下。

（1）所有接口都按照 RESTful API 风格设计，通过 HTTP 协议调用。

（2）所有微服务都按照以下 8 个接口封装其功能：

①创建服务。根据用户需求创建服务，服务承载着用户私有的一些使用设置，这些设置是一直存储，不会随着服务的关闭而重置（保存的信息可以是个性化场景配置，也可以是特定的素材），创建服务返回唯一的服务 ID 作为服务的标识。

②打开（运行）服务。给指定的服务申请对应的 IT 资源运行，进入可工作状态。接口参数可以按需加上回调 url，当服务运行过程中有需要通知业务方的可以通过此 url 进行，比如通知服务运行后的结果。打开（运行）服务返回服务当次运行期间有效凭证。

③关闭服务（可选）。服务当次使用结束，可以关闭服务来释放计算资源，减少资源的浪费。有些服务会自动关闭，不需要实现主动。

④获取服务信息。包括服务名称、当前运行状态（关闭、启动中的百分比、运行中）、资源占用情况（CPU、内存、IO 等运行时状态）等。

⑤销毁 / 删除服务。不再需要使用的服务可以删除，个性化配置将丢失。

⑥更新服务信息（可选）。服务创建完成后，可以修改创建时候提交的信息，如服务名称等。

⑦获取服务列表（可选）。可以获取某个用户下面的所有服务列表，包括简单的信息。

⑧服务设置（可选）。有些服务可以通过 API 方式设置，服务设置提供了默认设置不满足用户需求的情况

下，高级用户自行设置服务参数的接口。

该方案只需要对开发好的服务接口做简单的二次封装，功能边界清晰，开发成本低，易于实现。以上接口可以满足大部分微服务功能的表达，同时接口设计屏蔽微服务的具体功能，只考虑服务、用户和平台三者的关系进行抽象，降低平台管理复杂度，减轻用户学习成本。

5. 微服务模块镜像制作技术

容器云平台支持的两种镜像构建的方式，如图 4 所示。

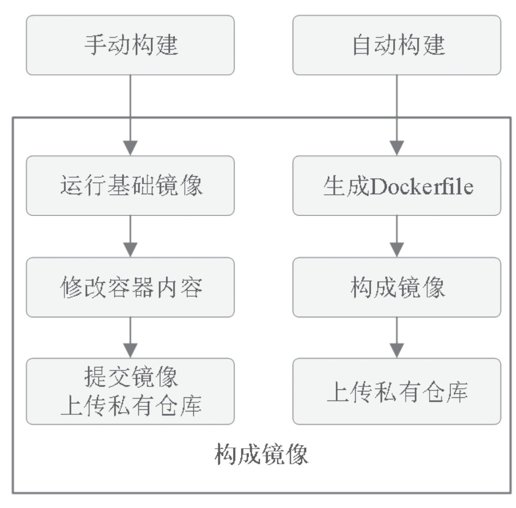


图 4　镜像构建的两种方式

其中自动构建可以根据用户选择的基础镜像，配置启动命令、环境变量、服务端口等相关参数后，后台会根据配置的参数生成 Docker file 文件，然后通过调用 Docker API 构建镜像，并上传到镜像仓库。

手动构建根据选择基础镜像创建容器，并提供 SSH 服务以及映射相关的服务端口，用户通过 SSH 登录容器后，修改配置文件，添加、删除文件等，且可以通过服务端口验证修改的情况，确认无误，并在容器云平台提交后，容器云平台会自动生成镜像并提交到仓库。容器云平台会记录每个用户的构建记录，通过云平台可以检索追踪历史构建记录，便于审计和分析。通过容器云平台的镜像管理控制台，可以浏览当前所有镜像，并支持多维度排序，比如按镜像名、用户名、构建时间等进行排序。同时支持在每个镜像的详情页，对镜像的 Logo、功能描述、应用程序端口、启动命令、环境变量、存储路径等进行重新编辑。也支持镜像使用频次的统计等数据的统计。对于已经部署的镜像，可以通过重新部署选择低版本的镜像进行回滚，这是在发布应用的时候经常使用的操作。构建成功对应服务的镜像后，在自己的 Docker 中运行该镜像即可实现服务的部署，然后通过 k8s 等对 Docker 进行管理即可对服务进行操作。

6. 总结与展望

视听媒体微服务第三方模块适配接入技术旨在拓展视听媒体微服务平台功能，规范微服务化划分原则，针对 API 网关的设计提出统一标准，实现除原生团队以外的第三方模块平滑接入到微服务平台，快速建立视听媒体微服务生态。

本文给出了对单体架构进行拆分、接口设计、容器化镜像制作的全流程解决方案。视听媒体微服务第三方模块适配接入技术有助于提升视听媒体微服务平台扩展性，增加平台功能，加速内容制作、提高内容质量。将第三方模块接入平台可以快速形成视听媒体制作生态，促进视听媒体行业发展。

随着研究内容的不断深入与细化，本文将选取不同的第三方模块进行试验验证。在试验验证的基础上，通过不断优化视听媒体微服务第三方模块适配接入技术，实现视听媒体微服务第三方模块适配接入解决方案，为形成第三方微服务模块接入测试技术规范打下基础。媒

参考文献

[1] 杨鸥，张昇，耿贞伟 . 微服务架构在容器云中的应用实践 [J]. 电脑与电信，2017（7）： 79-81.

[2] Jamshidi P，Pahl C，Mendonça NC，et al. Microservices： The journey so far and challenges ahead. IEEE Software，2018（3）：24-35.

[3] 黄显琛 . 基于微服务架构的系统设计与实现 [J]. 信息技术与信息化，2020（11）： 16-17.

[4] Richardson C. Pattern： Microservice architecture. 2018. http：//microservices. io/patterns/microservices.html

[5] Escobar D，Cárdenas D，Amarillo R，Castro E，Garcés K，Parra C，Casallas R. Towards the understanding and evolution of monolithic applications as microservices. In： Proc. of the 2016 XLII Latin American Computing Conf.（CLEI）. 2016. 1-11.

[6] 姚刚，吴海莉，王从镔 . 浅析微服务架构 API 网关的作用 [J]. 信息系统工程，2020（12）： 16-18.

[7] Ahmadvand M，Ibrahim A. Requirements reconciliation for scalable and secure microservice （de） composition. In： Proc. of the 2016 IEEE 24th Int'l Requirements Engineering Conf. Workshops （REW）. IEEE，2016：68-73.

[8] Levcovitz A，Terra R，Valente MT. Towards a technique for extracting microservices from monolithic enterprise systems. arXiv preprint arXiv：1605.03175，2016.

[9] Newman S，Wrote； Cui LQ，Zhang J，Translate. Building Microservices： Designing Fine-grained Systems. 2nd ed.，Beijing：People's Posts and Telecommunications Press，2016 （in Chinese）.

[10] Baresi L，Garriga M，Derenzis A. Microservices identification through interface analysis. In： Proc. of the European Conf. on Service-oriented and Cloud Computing. 2017：19-33.
[11] 龚方生 . 微服务中的 Docker 技术应用 [J]. 电子技术与软件工程，2021（4）： 54-56.
[12] 薛亮，侯婕 . 基于 Docker 的作战应用微服务化架构研究 [J]. 舰船电子工程，2020（3）： 22-25.
[13] Hassan S，Bahsoon R. Microservices and their design trade-offs： A self-adaptive roadmap. In： Proc. of the 2016 IEEE Int' l Conf. on Services Computing （SCC）. IEEE，2016：813-818.
[14] Evans E，Wrote； Zhao L，Sheng HY，Liu X，Translate. Domain-driven Design： Tracking Complexity in the Heart of Software.Beijing： People' s Posts and Telecommunications Press，2016 （in Chinese）.
[15] Kecskemeti G，Marosi AC，Kertesz A. The ENTICE approach to decompose monolithic services into microservices. In： Proc. of the 2016 Int' l Conf. on High Performance Computing Simulation （HPCS）. IEEE，2016：591-596.
[16] Hassan S，Ali N，Bahsoon R. Microservice ambients： An architectural meta-modelling approach for microservice granularity. In： Proc. of the 2017 IEEE Int' l Conf. on Software Architecture （ICSA）. IEEE，2017：1-10.
[17] Gysel M，KoLbener L，Giersche W，et al. Service cutter： A systematic approach to service decomposition. In： Proc. of the European Conf. on Service-oriented and Cloud Computing. Springer-Verlag，2016： 185-200.
[18] Mazlami G，Cito J，Leitner P. Extraction of microservices from monolithic software architectures. In： Proc. of the 2017 IEEE Int' l Conf. on Web Services （ICWS）. 2017：524-531.

作者简介：付坤（1999-），男，湖北孝感，学生，硕士，研究方向：微服务相关技术，容器技术；邓成胜（1998-），男，重庆，研究生，研究方向：微服务相关技术，大数据应用；倪业鹏（1984-），男，宁夏银川，高级工程师，工学博士，研究方向：视听媒体微服务技术，融合媒体评测技术。

（责任编辑：张晓婧）

（上接第124页）

版与网络营销整合工作逐渐呈现新的信息传播思维模式，媒体资源融合时代背景下的网络编辑不仅需要做好网络内容的资源整合与营销编辑管理工作，同样还需要能够充分利用移动互联网资源整合营销部和出版社的资源，进一步不断扩大网络图书内容出版的社会影响力和范围，提高网络内容出版产品的社会影响力。整合内容出版资源与大众出版编辑资源，在把握掌控中国大众出版阅读消费心理的良好基础上迅速占据国内出版市场先机。[7]

结语

综上所述，在媒介融合的背景下，科技图书编辑需要充分意识到媒介融合背景下科技图书编辑发展面临的挑战，探索编辑的转型渠道，提高产品更新的效率，关注选题的趣味性、人工智能技术，改变思维方式，提高资源整合能力。同时，要跟上时代的发展步伐，充分提高竞争力，促进传统的科技书籍在媒体融合背景下完成快速转型。媒

参考文献

[1] 赵春霞 . 新媒体时代图书策划编辑工作思路探析 [J]. 中国战略新兴产业（理论版），2019（18）：1.
[2] 方艳. 浅谈出版融合背景下科技类图书编辑的角色定位 [J]. 科学与信息化，2019（25）：193-195.
[3] 王爱荣 . 媒介融合背景下的文化创意产业者素养提升——以现代出版图书编辑为例 [J]. 科技传播，2019（16）：153-154.
[4] 谭彩霞，湛江 . 新媒体融合时代我国图书出版的现状及对策分析 [J]. 金陵科技学院学报，2020（1）：44-47.
[5] 肖晗，吕先竞，邓英，燕朝西，李秀燕，杨瑜 . 媒体融合背景下学术期刊影响力提升机制探讨 [J]. 西南石油大学学报（社会科学版），2019（4）：104-109.
[6] 刘晓敏 . 新媒体时代图书出版编辑工作创新路径刍议 [J]. 研究导刊，2019（19）：188-195.
[7] 严冬芳 . 媒介融合背景下传统媒体编辑的转型分析 [J]. 西部广播电视，2019（23）：171-172.

作者简介：刘燕芳（1987-），女，河南开封，硕士，编辑，研究方向：科技类图书编辑。

（责任编辑：张晓婧）